# Wprowadzanie nadmiaru informacyjnego dla transmitowanych szeregowo danych w kompaktowych sterownikach PLC firmy Mitsubishi Electric

Roman Mielcarek

**Streszczenie:** W artykule przedstawiono trzy metody programowego kodowania nadmiarowego danych, mogące mieć zastosowanie w protokołach użytkownika implementowanych w sterownikach PLC typu FX firmy Mitsubishi Electric. Rozpatrzono przykłady kodów definiowanych układem równań, kodów tablicowych i kodów w zapisie wielomianowym. Dla każdej z rozpatrywanych metod kodowania przedstawiono fragmenty przykładowych programów realizujących te metody w sterowniku FX.

## 1. Wprowadzenie

W sterownikach PLC typu FX firmy Mitsubishi Electric [4, 5] istnieje możliwość bardzo łatwego szeregowego transferu danych, opartego na wbudowanych protokołach transmisyjnych oraz protokołach własnych – protokołach użytkownika. Możliwość ta bazuje na otwartym programowo dostępie do konfiguracji własnych (wbudowanych w sterownik) portów szeregowych sterownika, jak i dodatkowych portów szeregowych, zawartych w dołączonych dodatkowo modułach komunikacyjnych. W najszybszym sterowniku typu FX3U do dyspozycji użytkownika są dwa wbudowane porty szeregowe, w pozostałych typach – jeden. Porty wbudowane można wyposażyć w jeden z trzech standardowych styków transmisyjnych typu RS o numeracji 232, 422 i 485. Ta dostępność sprzętowa i programowa daje szerokie możliwości zaimplementowania własnego protokołu transmisyjnego (protokołu użytkownika) [2, 5], jeśli protokoły wbudowane z jakichś względów nie spełniają wymogów dla danej aplikacji. Zastosowanie protokołu użytkownika obliguje koniecznością wprowadzenia po stronie nadawczej dodatkowych danych, nazywanych dalej nadmiarem informacyjnym, które służą do weryfikacji danych użytkownika po stronie odbiornika. Proces ten nazywany jest kodowaniem nadmiarowym danych po stronie nadawczej i dekodowaniem nadmiarowym po stronie odbiorczej. Celem tego procesu jest zapobieżenie (z określonym prawdopodobieństwem) przekazania błędnych danych (zmienionych w trakcie transmisji na skutek zakłóceń w kanale transmisyjnym) do obszaru ich przeznaczenia po stronie odbiorczej.

W sterownikach FX istnieje kilka instrukcji zaawansowanych [3] ogólnego przeznaczenia, które można wykorzystać w procesie programowego kodowania, jak i dekodowania danych. Oprócz nich sterowniki te posiadają również kilka instrukcji specjalizowanych, przydatnych w wybranej metodzie kodowania, które zostaną przedstawione przy rozpatrywaniu danej metody kodowania.

## 2. Podstawy teoretyczne kodowania nadmiarowego

Zastosowanie własnego lub wbudowanego sposobu kodowania danych wymaga zrozumienia przez użytkownika podstaw teorii kodowania nadmiarowego. Zasadniczą klasą kodów są kody liniowe, przedstawiane w zapisie macierzowym (układu równań) lub zapisie wielomianowym, oraz kody nieliniowe typu „suma kontrolna”. Stąd też poniżej zostanie pokrótce omówiona każda z tych klas.

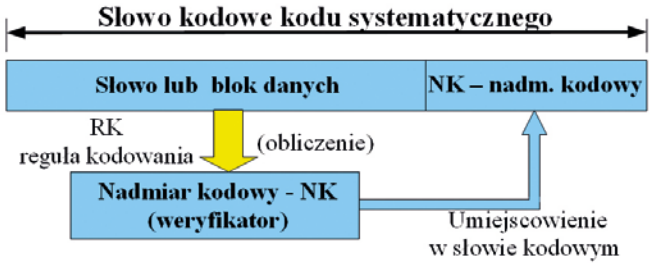
### 2.1. Kody liniowe w zapisie macierzowym

Kodem nadmiarowym nazywana będzie dalej taka struktura danych, w której do bloku $k$ bitów danych dodane zostanie w procesie kodowania $r$ bitów nadmiarowych. Postać (określenie kombinacji) tych $r$ bitów nadmiarowych określa przyjęta reguła kodowania. Zakłada się również, że te $r$ bitów nadmiarowych zależeć będzie tylko od aktualnego bloku tych $k$ bitów danych, a nie od danych, które zostały wcześniej nadane. Tak więc blokowy kod nadmiarowy można scharakteryzować tymi dwoma parametrami liczbowymi i zapisać w postaci $(k+r, k) = (n, k)$, gdzie liczba $n$ określa sumę bitów danych i bitów nadmiarowych. Ten blok $n$ bitów przedstawiany będzie w postaci $n$-bitowego słowa, zwanego słowem kodowym. Jeśli przyjąć, że bity danych (zwane również pozycjami informacyjnymi) występują na początku tego słowa kodowego (numerując od jego lewej strony), a pozycje nadmiarowe (wprowadzone w procesie kodowania) są końcowymi pozycjami tego słowa, to tak określony kod nazywany będzie dalej kodem systematycznym. Sposób określania pozycji nadmiarowych w kodzie systematycznym przedstawiono na rys. 1.

Wśród możliwych reguł kodowania określających pozycje nadmiarowe największą popularność zdobyły kody liniowe, definiowane układem równań z operacją liniową w postaci sumy mod. 2 elementów bitowych słowa kodowego [1]. Układ tych równań przedstawiono w syntetycznym zapisie we wzorze (1).

$$\sum_{j=1}^{n} Sj * Hjl = 0 \qquad dla\ l = 1, 2, \ldots, r \tag{1}$$

gdzie:
$\sum$ - operator wieloargumentowej sumy mod.2
$Sj$ - j-ta pozycja słowa kodowego
$Hjl$ - współczynnik {0,1} określający przynależność pozycji Sj do l-tego równania




Rys. 1. Proces tworzenia słowa kodu systematycznego

$$H = \begin{bmatrix} 1000\,0111 & 1000\,0000 \\ 1100\,0011 & 0100\,0000 \\ 1110\,0001 & 0010\,0000 \\ 1111\,0000 & 0001\,0000 \\ 0111\,1000 & 0000\,1000 \\ 0011\,1100 & 0000\,0100 \\ 0001\,1110 & 0000\,0010 \\ 0000\,1111 & 0000\,0001 \end{bmatrix} \qquad G = \begin{bmatrix} 1000\,0000 & 1111\,0000 \\ 0100\,0000 & 0111\,1000 \\ 0010\,0000 & 0011\,1100 \\ 0001\,0000 & 0001\,1110 \\ 0000\,1000 & 0000\,1111 \\ 0000\,0100 & 1000\,0111 \\ 0000\,0010 & 1100\,0011 \\ 0000\,0001 & 1110\,0001 \end{bmatrix}$$

$$H_{r\times n} = [P_{r\times k}, I_{r\times r}] \qquad G_{k\times n} = [I_{k\times k}, P^T_{k\times r}]$$

Rys. 2. Macierz kontrolna *H* i macierz generująca *G* przykładowego kodu (*n*, *k*) = (16,8)

W powyższym układzie równań numeracja pozycji danych odbywa się dla $1 \le j \le k$, gdzie pozycja $j=1$ wskazuje zazwyczaj najstarszą (w numeracji systemowej) pozycję słowa danych. Pozostałe numery od $k+1$ do $n$ określają pozycje nadmiarowe. W kodzie systematycznym każde równanie powyższego układu zawiera tylko jedną pozycję nadmiarową. Współczynniki *Hjl* zapisane w postaci tablicy, tak jak występują w kolejnych równaniach, tworzą tzw. macierz kontrolną kodu *H*. Bierze ona swą nazwę od pozycji kontrolnych – nadmiarowych, które definiuje. Przykład macierzy kontrolnej dla rozpatrywanego dalej kodu o parametrach (*n*, *k*) = (16,8) przedstawiono na rys. 2.

Zbiór słów kodowych kodu liniowego określony według równań (1) tworzy strukturę algebraiczną zwaną grupą. Struktura ta spełnia pięć aksjomatów [1], z których dwa najważniejsze dla teorii kodowania liniowego są następujące:

- A1 – grupa zawiera słowo zerowe *Sz* (słowo złożone z samych zer – słowo neutralne);
- A2 – suma mod. 2 dwóch dowolnych słów kodowych (suma po odpowiadających sobie pozycjach) jest również słowem kodowym ze zbioru grupy.

Aksjomat *A1* określa warunek konieczny przynależności danego zbioru słów kodowych do klasy kodów liniowych, natomiast aksjomat *A2* (aksjomat zamkniętości) pozwala określić zbiór słów kodowych kodu liniowego na bazie podzbioru słów cechujących się tzw. niezależnością liniową. Podzbiór taki ma tę własność, że dowolnego słowa z tego podzbioru nie da się wygenerować z pozostałych w oparciu o aksjomat *A2*. Podzbiór ten, zapisany w postaci tablicy, nosi nazwę macierzy generującej *G*. Macierze *H* oraz *G* dla przykładowego kodu o parametrach (*n*, *k*) = (16,8) przedstawiono na rys. 2. Parametr $r = n–k$ określa liczbę pozycji kontrolnych w słowie kodowym.

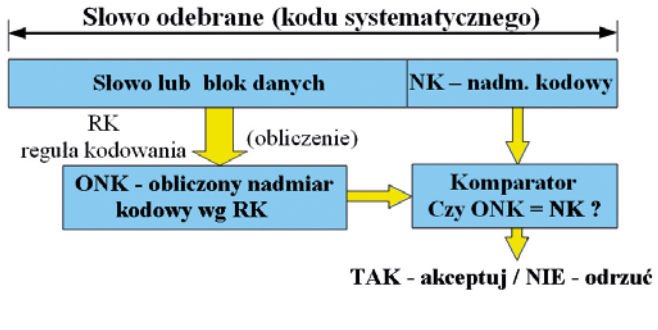
Z powyższego przykładu wynika, że kod liniowy może być definiowany dwojako: za pomocą macierzy *H* (układu równań) lub za pomocą macierzy *G* (podzbiorem liniowo niezależnych słów kodowych). Oba sposoby definiowania kodu (obie macierze) łączy podmacierz *P* występująca w postaci prostej w macierzy *H* i transponowanej w macierzy *G*.

Wprowadzone pozycje nadmiarowe (kontrolne) w nadajniku, służą do weryfikacji w odbiorniku odebranego słowa w oparciu o zastosowaną regułę kodowania. Proces kontroli zastosowanej reguły kodowania (proces dekodowania) przedstawiono na rys. 3.

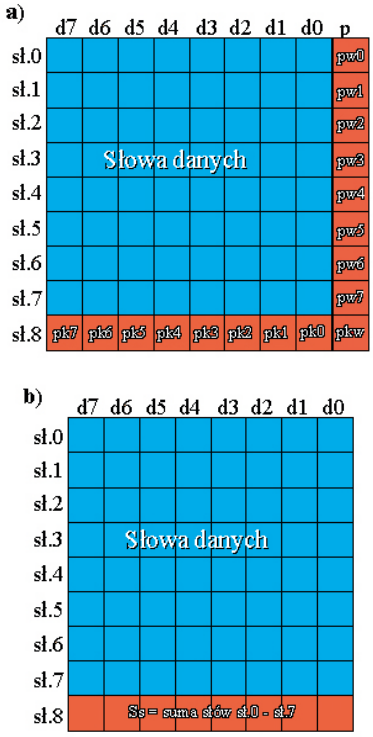
Podjęcie decyzji NIE w dekoderze (komparatorze) odbiornika ma miejsce w przypadku wykrycia błędu w słowie odebranym, czyli stwierdzenia niezgodności w przyjętej regule kodowania, co ma miejsce w przypadku wystąpienia tzw. błędu wykrytego w procesie transmisji. Decyzja TAK oznacza zgodność obliczonego nadmiaru kodowego ze stosowaną regułą kodowania. Ta zgodność występuje w dwóch przypadkach:

- gdy odebrane słowo jest bezbłędne – identyczne, jak zostało nadane w nadajniku;
- gdy wystąpił tzw. błąd niewykryty – nadane słowo kodowe zostało przekłamane w inne słowo kodowe, które z definicji spełnia regułę kodowania.

Problem tworzenia reguły kodowania sprowadza się przede wszystkim do uzyskania maksymalnej liczby błędów wykrytych w zbiorze wszystkich możliwych błędów w danym kodzie (*n*, *k*).




Rys. 3. Proces dekodowania odebranego w odbiorniku słowa




Rys. 4. Przykłady kodów tablicowych: a) kod podwójnie iterowany; b) kod z sumą kontrolną

### 2.2. Kody w zapisie tablicowym

Kodowanie wg układu równań przedstawione w p. 2.1. kojarzy się zazwyczaj z uzupełnieniem pojedynczego słowa systemowego o słowo nadmiarowe. Jednak w większości systemów transmisyjnych jednorazowemu kodowaniu nadmiarowemu podlega większa liczba słów (np. bajtów) danych. Mogą one być wówczas przedstawione w tablicy o wymiarach *m*-wierszy i *l*-kolumn, a każdy wiersz i kolumna mogą być uzupełnione o pozycje nadmiarowe dowolnego kodu. Otrzymany wówczas kod nazywany jest kodem podwójnie iterowanym (tablica ma dwa wymiary). Kodowanie wierszy tablicy odbywa się zazwyczaj za pomocą tzw. bitu parzystości (lub nieparzystości), który podczas nadawania z nadajnika słowa danych obliczany i dodawany jest układowo na końcu jego transmisji w sposób automatyczny. Natomiast kodowanie kolumn odbywa się programowo. W trakcie nadawania kolejnych słów danych (wierszy tablicy) obliczana jest suma wzdłużna, np. w postaci sumy mod. 2 poszczególnych pozycji (kolumn) tych słów, która po zakończeniu słów tablicy nadawana jest jako ostatnie słowo bloku kodowego. Słowo sumy wzdłużnej może być również obliczane jako suma arytmetyczna nadawanych słów danych. Metoda ta jest stosowana we wszystkich protokołach wewnętrznych sterowników FX jako zabezpieczenie transmitowanego bloku danych [5].

Na rysunku 4 przedstawiono dwie tablice, obrazujące dwa sposoby kodowania tablicowego. Na rys. 4a przedstawiono typowy liniowy kod iterowany, który posiada parametry ($n$, $k$) = (80, 64). Kodowanie odbywa się wg powyżej przedstawionego opisu. Natomiast na rys. 4b przedstawiono nieliniowy (wg definicji z rys. 2) kod tablicowy oparty na bajcie arytmetycznej sumy kontrolnej, obliczanej jako suma nadawanych słów, traktowanych jako liczby 8-bitowe. Jeśli tę tablice uzupełni się o kolumnę bitu parzystości wzdłużnej (jak na rys. 4a), to uzyska się kod o lepszych własnościach detekcyjnych (własnościach wykrywania błędów) niż kod z rys. 4a.

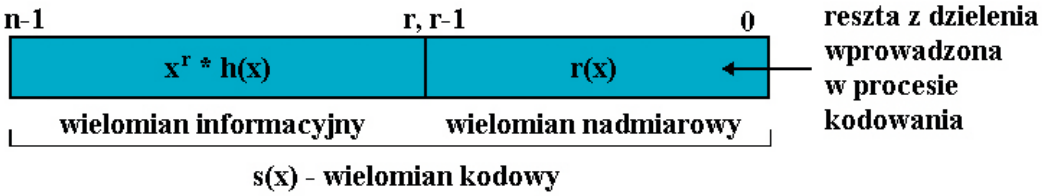
### 2.3. Kody w zapisie wielomianowym

Konieczność wprowadzenia większej niż jeden liczby bitów nadmiarowych dla słowa informacyjnego o długości większej niż słowo systemowe napotyka duże trudności aplikacyjne. Przyczyną jest złożone wydzielanie danego bitu informacyjnego do danego równania (wg 1), określającego daną pozycję nadmiarową. Stąd też zastosowanie kodu o lepszych własnościach detekcyjnych (kodu z większą liczbą pozycji nadmiarowych) według metody macierzowej nie jest stosowane. Rozwiązaniem jest zastosowanie kodowania i dekodowania wielomianowego, którego proces odbywa się w trakcie samego nadawania i odbioru słowa kodowego, czyli „w biegu". Układem realizującym te procesy jest rejestr szeregowy ze sprzężeniem zwrotnym, który pobiera dane z wyjścia rejestru nadawczego nadajnika. Kody te znalazły najszersze zastosowanie w zabezpieczaniu transmitowanych danych z powodu prostoty kodowania i dekodowania oraz z uwagi na stosunkowo małą złożoność układową czy programową procesu kodowania i dekodowania.

Kod wielomianowy uzyskuje się po wprowadzeniu operacji mnożenia binarnego do definiowania zbioru grupy słów kodowych, przez co uzyskuje się strukturę algebraiczną zwaną pierścieniem. Struktura ta posiada wszystkie atrybuty grupy, lecz elementy zbioru grupy (słowa kodowe) muszą spełniać dodatkowo tę nowo wprowadzoną własność, czyli być podzielne przez określony czynnik. Wprowadzenie operacji mnożenia do zbioru słów kodowych wiąże się z przedstawieniem tych słów w postaci wielomianu o współczynnikach binarnych. Wartość tych współczynników stanowi kombinację słowa kodowego. Poniżej przedstawiono konwersję zapisu przykładowego 5-bitowego słowa w postać wielomianową, gdzie znak ⊕ jest operatorem mod. 2.

$$H = 10\,111 \;<=>\; h(x) = \\ = 1{*}x^4 \oplus 0{*}x^3 \oplus 1{*}x^2 \oplus 1{*}x^1 \oplus 1{*}x^0 = x^4 \oplus x^2 \oplus x \oplus 1 \qquad (2)$$

Do wygenerowania kodu o parametrach ($n$, $k$) należy określić wielomian stopnia $r = n–k$, który nazywany jest wielomianem generującym $G(x)$. Wielomian ten reprezentuje jedno wybrane słowo kodowe, na bazie którego są generowane wszystkie




$$\frac{x^r * h(x)}{g(x)} = A(x) \oplus \frac{r(x)}{g(x)} * g(x) \Rightarrow A(x) * g(x) = x^r * h(x) \oplus r(x)$$

$$A(x) * g(x) \in \{S\} \Rightarrow x^r * h(x) \oplus r(x) \in \{S\}$$

Rys. 5. Proces tworzenia wielomianowego kodu systematycznego

pozostałe słowa kodowe. Każdy wielomian kodowy $s(x)$ jest podzielny bez reszty przez wielomian generujący $g(x)$, co można określić następującym równaniem:

$$s(x) \bmod g(x) = 0 \qquad (3)$$

W procesie kodowania wielomianowego, dla kodowania rozdzielnego operującego na słowie kodu systematycznego, podstawową rolę odgrywa operacja dzielenia wielomianów. Jest ona konieczna miedzy innymi do sprawdzenia równania (3).

Proces tworzenia wielomianu kodowego w konwencji rozdzielnej – słowa kodu systematycznego – przedstawiono na rys. 5. Pozycje słowa numerowane są odwrotnie niż w zapisie macierzowym, czyli tak jak numerowane są pozycje słowa systemowego np. w sterowniku PLC. Istotą tego procesu jest takie określenie kombinacji pozycji nadmiarowych (zakres o 0 do $r$–1), aby całe słowo kodowe $s(x)$ (wielomian kodowy) spełniało równanie (3).

W dolnej części rysunku 5 podano matematyczny zapis sposobu określenia części nadmiarowej $r(x)$ wielomianu kodowego $s(x)$. Wynika z niego, że $r(x)$ jest resztą z podzielenia wielomianu informacyjnego $h(x)$ przesuniętego o $r$ pozycji w lewo (pomnożonego przez $x^r$).

Jak wynika z dowodu na rys. 5, w procesie kodowania wielomianowego (w dekodowaniu również) dla kodu rozdzielnego podstawową rolę odgrywa operacja dzielenia wielomianów. Przykładowy proces dzielenia dwóch wielomianów $y(x)$ i $g(x)$ w postaci:

- dzielnej: $y(x) = x^7 \oplus x6 \oplus x^5 \oplus x^4 \oplus x \oplus 1$;
- dzielnika: $g(x) = x^3 \oplus x \oplus 1$;

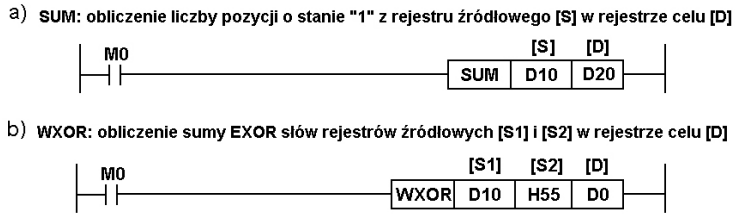
przedstawiono na rys. 6. W ostatnim wierszu znajduje się reszta $r(x)$ niepodzielna przez $g(x)$.

### 3. Kodowanie i dekodowanie w sterowniku FX

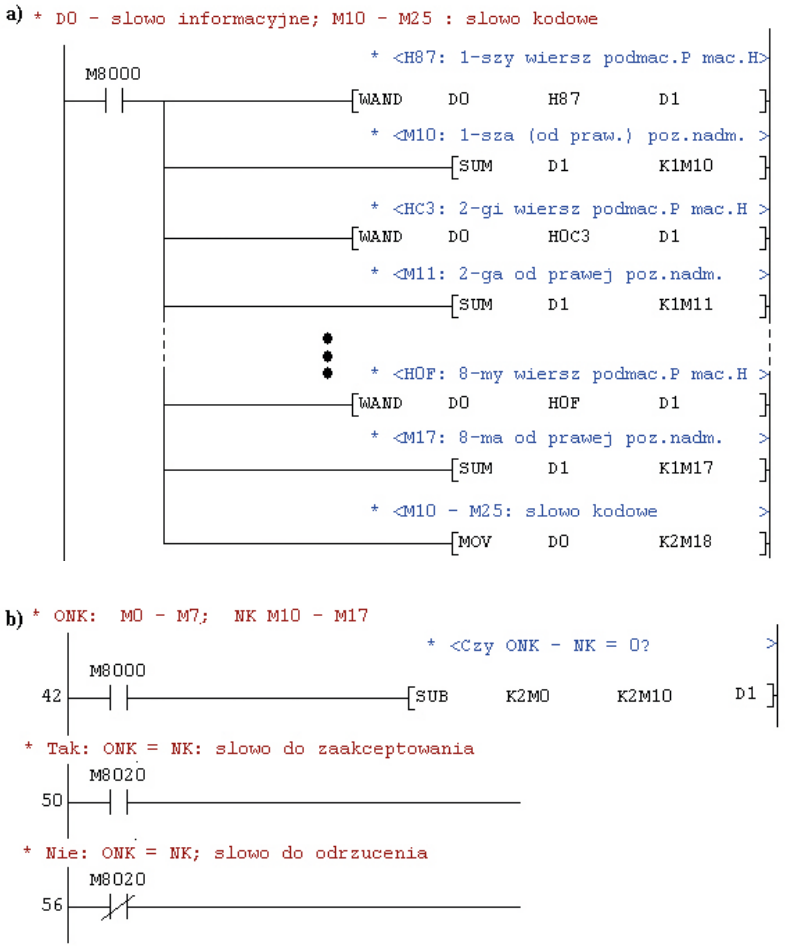
Każdą z opisanych wyżej metod kodowania można zaimplementować w sterownikach FX. Przedstawione w kolejnych punktach metody zapisane zostały w języku drabinkowym tego sterownika w notacji programu narzędziowego GX Developer. Każdy ze szczebli tego programu (patrz rys. 7) zaczyna się zestykiem stanu urządzenia, który przyjmuje wartość logiczną 0 lub 1 (ang. *OFF* lub *ON*). Zestyk ten jest jednocześnie zestykiem warunku wykonania instrukcji zaawansowanej, której skrót (mnemonik) wraz z argumentami jest zapisywany w nawiasach prostokątnych na końcu linii szczebla. Warunek „1” (*ON*) powoduje wykonanie instrukcji zaawansowanej,

| $x^7$ | $x^6$ | $x^5$ | $x^4$ | $x^3$ | $x^2$ | x | 1 | <= Pozycje dzielnej y(x) | Składnik ilorazu h(x) | Dzielnik g(x) |
|---|---|---|---|---|---|---|---|---|---|---|
| $x^7$ | $x^6$ | $x^5$ | $x^4$ | 0 | 0 | x | 1 | Dzielna | | |
| $x^7$ | 0 | $x^5$ | $x^4$ | 0 | 0 | 0 | 0 | s(x) : g(x) | $x^4$ | |
| 0 | $x^6$ | 0 | 0 | 0 | 0 | x | 1 | $= s_1(x)$ | | |
| | x6 | 0 | $x^4$ | $x^3$ | 0 | 0 | 0 | $s_1(x) : g(x)$ | $x^3$ | |
| | 0 | 0 | $x^4$ | $x^3$ | 0 | x | 1 | $= s_2(x)$ | | $x^3 \oplus x \oplus 1$ |
| | | | $x^4$ | 0 | $x^2$ | x | 0 | $s_2(x) : g(x)$ | x | |
| | | | 0 | $x^3$ | $x^2$ | 0 | 1 | $= s_3(x)$ | | |
| | | | | $x^3$ | 0 | x | 1 | $s_3(x) : g(x)$ | 1 | |
| | | | | 0 | $x^2$ | x | 0 | = r(x) | | |

Rys. 6. Proces dzielenia przykładowych wielomianów y(x) przez g(x)




Rys. 7. Instrukcje sterownika FX stosowane w kodowaniu i dekodowaniu macierzowym




Rys. 8. Fragmenty przykładowych programów: a) kodowania według macierzy *H*; b) dekodowania

warunek „0” (*OFF*) powoduje pominięcie tej instrukcji. Wśród argumentów instrukcji zaawansowanych wyróżnić można argumenty: źródłowe [S], parametryczne – liczbowe [n] i przeznaczenia [D]. Argumentami [S] i [D] są urządzenia sterownika (np. rejestry, liczniki, zespoły kolejnych urządzeń bitowych itp).

### 3.1. Kodowanie macierzowe w sterownikach FX

Zastosowanie kodowania macierzowego za pomocą macierzy kontrolnej H lub generującej G ma aspekt czysto akademicki i nie jest stosowane praktycznie. Służyć może tylko ugruntowaniu wiedzy z zakresu teorii kodowania i zasad działania sterownika PLC. Tylko z tego też względu zagadnienie to zostanie przedstawione poniżej.

Zastosowanie tego rodzaju kodowania (jak również wszystkich pozostałych) wymaga przyjęcia założenia, że liczba bitów nadmiarowych jest zawsze pełną wielokrotnością długości systemowego słowa danych. Dla wartości $r = k$ prowadzi to do kodu o parametrach $(n, k) = (2k, k)$, dla którego obie macierze *H* i *G* są tych samych rozmiarów jak na rys. 1, co jest przypadkiem szczególnym. Przyjęcie warunku $r < k = 8$ implikuje transmisję słowa nadmiarowego z pustymi, niewykorzystanymi bitami.

Obliczenie danej pozycji nadmiarowej wg równań z rys. 2 wymaga obliczenia sumy mod. 2 z wybranych pozycji słowa danych. Sterowniki FX pozwalają na takie obliczenie metodą pośrednią z wykorzystaniem instrukcji SUM (*Sum of active bits*), przedstawionej w języku drabinkowym na rys. 7a. Instrukcja ta, jeśli warunek wykonania M0 = „1”, dokonuje sumowania wszystkich 16 pozycji rejestru źródłowego [S] = D10 o wartości „1” i umieszcza wynik sumowania w rejestrze celu [D] = D20. Najmłodszy bit rejestru celu jest zatem bitem parzystości (0) lub nieparzystości (1) liczby pozycji „1” w słowie źródłowym. Jest więc poszukiwaną wartością danej pozycji nadmiarowej.

Fragment przykładowego programu drabinkowego, obliczającego słowo kodowe w markerach M25 ÷ M10 w oparciu o macierz H z rys. 2, przedstawiono na rys. 8a. Osiem kolejnych instrukcji SUM oblicza kolejne pozycje nadmiarowe, pozostawione kolejno w markerach M10 ÷ M17. Zapis KnMm [3÷5] oznacza rejestr (słowo, liczbę) kolejnych urządzeń bitowych o długości n*4

bitów, rozpoczynający się od markera M o numerze m.

Proces dekodowania odebranego słowa przedstawiono na rys. 8 b. Przebiega on podobnie. Najpierw jest obliczany nadmiar kodowy ONK ze słowa informacyjnego słowa odebranego (tak jak na rys. 8 a), a następnie jest on porównywany ze słowem nadmiarowym NK słowa odebranego. Porównanie (rys. 3) odbywa się metodą odejmowania obu liczb, które reprezentują oba słowa nadmiarowe. Wynik porównania znajduje się w systemowym rejestrze zera operacji arytmetycznej M8020.

Program obliczający nadmiar kodowy wg macierzy generującej G jest jeszcze prostszy, gdyż składa się w rozpatrywanym przykładzie tylko z ośmiu instrukcji WXOR, obliczających słowo nadmiarowe na podstawie bitów „1" słowa nadmiarowego. Argument bezpośredni instrukcji WXOR sumowany z tworzonym słowem nadmiarowym (K2M18) określa kolejny wiersz podmacierzy P macierzy G.

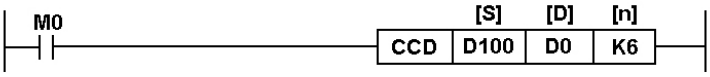
### 3.2. Kodowanie i dekodowanie tablicowe w sterownikach FX

Jak wspomniano w punkcie 2.2. w większości systemów transmisyjnych jednorazowemu kodowaniu nadmiarowemu podlega większa od jedności liczba bajtów (słów) danych. Mogą one być wówczas wbudowane w tablicę o wymiarach *m*-wierszy i *l*-kolumn, a każdy wiersz i kolumna mogą być uzupełnione o pozycje nadmiarowe dowolnego kodu. Parametr l wynosi zazwyczaj wartość 8 (większość systemów transmisyjnych posługuje się transmisją bajtową) lub jest jego wielokrotnością. Długość nadmiaru wzdłużnego i poprzecznego przyjmuje zazwyczaj wartość 1.

Wprowadzenie bitu kontrolnego do każdego nadawanego słowa danych przez dodatkowe porty szeregowe sterownika FX [5] deklaruje się w ich rejestrach konfiguracyjnych D8120 oraz D8400 i D8420 (FX3U). Natomiast poprawność tego bitu po stronie odbiornika sygnalizowana jest w rejestrach D8063 i D8438 (FX3U).

Obliczenie słowa parzystości wzdłużnej lub sumy kontrolnej można wykonać za pomocą instrukcji WXOR lub ADD, lub też za pomocą jednej specjalizowanej do tego celu instrukcji CCD (*check code*) [4], której sposób działania prezentują tabele na rys. 9. Stan markera systemowego M8161 definiuje postać bloku danych (również dla procesu nadawania i odbioru), dla którego obliczane są obie sumy. Blok ten składać się może albo z [n/2] słów sterownika dla M8161 = „0", albo z młodszych bajtów kolejnych [n] słów bloku dla wartości M8161 = „1". Po wykonaniu instrukcji CCD wybrany rejestr wyniku jest nadawany za blokiem danych.

**CCD: obliczenie sumy arytmetycznej i sumy parzystości bajtów w rejestrach celu [D] i [D+1] ze stosu [n] słów o adresie początkowym [S]**




| M8161 = „0" | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Źródło (S) | | | Kombinacja bajtu | | | | | | | |
| D100 | H | FF | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | L | FF | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| D101 | H | FF | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | L | 00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| D102 | H | F0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| | L | 0F | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| Parzystość pionowa: D1 | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Suma: D0 | | | 3FC | | | | | | | |

| M8161 = „1" | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Źródło (S) | | Kombinacja bajtu | | | | | | | |
| D100 L | FF | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| D101 L | 00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| D102 L | 0F | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| D103 L | F0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| D104 L | F0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| D105 L | 0F | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| Parzystość pionowa: D1 | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Suma: D0 | | 2FD | | | | | | | |

Rys. 9. Ilustracja działania instrukcji CCD sterownika FX obliczającej sumę wzdłużną

Dekodowanie odebranego bloku kodowego jest analogiczne jak dla nadawanego bloku danych. Najpierw na bajtach bloku danych należy wykonać instrukcję CCD, określając ONK, a następnie należy porównać go z odebranym nadmiarem kodowym bloku NK za pomocą programu jak na rys. 8 b.

### 3.3. Operacja dzielenia wielomianowego w sterownikach FX

Z analizy procesu dzielenia przedstawionego na rys. 6 wynika, że proces ten realizowany jest na pozycjach binarnych $x^i$ słowa dzielnej. Oznacza to, że kombinację dowolnego wielomianu o współczynnikach binarnych można bezpośrednio przenieść do kolejnych urządzeń bitowych sterownika, z których najbardziej do tego celu nadają się znaczniki pomocnicze – markery. Na słowie utworzonym z tych kolejnych urządzeń można wykonywać operacje sumowania mod. 2, wykorzystując przedstawioną na rys. 7 instrukcję WXOR.

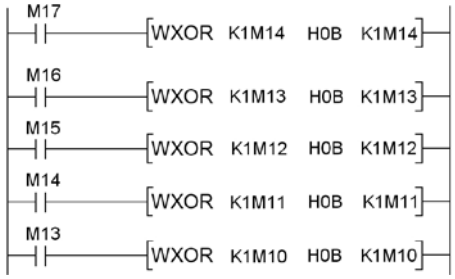
Operacja dzielenia przedstawiona na rys. 6 wymaga wpisania wielomianu dzielnej $y(x)$ do rejestru kolejnych markerów, np. w zakresie M10÷M17, i wykonania prostego programu dzielenia, czyli sumowania mod. 2. Dwie wersje programów dzielenia dla rozpatrywanego przypadku przedstawiono na rys. 10 a i 10 b. W pierwszej z nich (rys. 10 a) dzielnik jest równy zmiennej (zmniejszającej się) długości dzielnej dla każdego etapu dzielenia (sumowania mod. 2), a jego najmłodsza pozycja jest zawsze w markerze M10. W drugiej (rys. 10 b) dzielnik ma stałą pierwotną długość, lecz przypisywany jest od kolejnego młodszego markera w kolejnym etapie dzielenia – sumowania mod. 2 z dzielną.

Działanie każdego z tych dwóch wersji programu zilustrowano tabelą stanu markerów po wykonaniu kolejnego sumowania mod. 2 (instrukcji WXOR). I tak tabela na rys. 10 c ilustruje metodę z dzielnikiem o zmiennej długości (program z rys. 10 a), natomiast tabela na rys. 10 d przestawia metodę z dzielnikiem o stałej długości. W pierwszej z nich każda programowa instrukcja warunkowa WXOR ma inny drugi operand źródłowy (dzielnik zapisany w kodzie HEX), natomiast w drugiej operand ten ma stałą kombinację, a zmienia się tylko zakres markerów, z którymi jest on sumowany mod. 2. Efektem każdego programu jest reszta z dzielenia $r(x)$, znajdująca się po wykonaniu programu w markerach M10÷M12.

Przytoczone na rysunkach 10 a i 10 b programy służą do tworzenia słów kodowych dla kodu o parametrach $(n, k) = (8, 5)$, z wielomianem generującym $g(x) = x^3 \oplus x \oplus 1$. Aby wygenerować słowo kodowe tego kodu, należy 5-bitowe słowo danych źródłowych $h(x)$ wpisać do markerów M13÷M17 (przy zerowych wartościach M10÷M12) i wykonać daną wersję programu. Poszukiwana reszta pozostanie w markerach M10÷M12. Następnie należy ponownie

a)


b)




c)

| | $x^7$ | $x^6$ | $x^5$ | $x^4$ | $x^3$ | $x^2$ | x | 1 | <= Pozycje dzielnej | Składnik ilorazu h(x) |
|---|---|---|---|---|---|---|---|---|---|---|
| | M17 | M16 | M15 | M14 | M13 | M12 | M11 | M10 | <= Markery Mi dzielnej | |
| | $x^7$ | $x^6$ | $x^5$ | $x^4$ | 0 | 0 | x | 1 | Dzielna s(x) | |
| WXOR | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | Stan Mi | |
| | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | s(x) : H0B0 | $x^4$ =M17=1 |
| = WXOR | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | = $s_1$ (x) | |
| | | 1 | 0 | 1 | 1 | 0 | 0 | 0 | $s_1$ (x) : H58 | $x^3$ =M16=1 |
| = WXOR | | 0 | 0 | 1 | 1 | 0 | 1 | 1 | = $s_2$ (x) | |
| | | | | 1 | 0 | 1 | 1 | 0 | $s_2$ (x) : H16 | $x^1$ =M14=1 |
| = WXOR | | | | 0 | 1 | 1 | 0 | 1 | = $s_3$ (x) | |
| | | | | | 1 | 0 | 1 | 1 | $s_3$ (x) : H0B | $x^0$ =M13=1 |
| = | | | | | 0 | 1 | 1 | 0 | = r(x) | |

d)

| | $x^7$ | $x^6$ | $x^5$ | $x^4$ | $x^3$ | $x^2$ | x | 1 | <= Pozycje dzielnej | Składnik ilorazu h(x) |
|---|---|---|---|---|---|---|---|---|---|---|
| | M17 | M16 | M15 | M14 | M13 | M12 | M11 | M10 | <= Markery Mi dzielnej | |
| | $x^7$ | $x^6$ | $x^5$ | $x^4$ | 0 | 0 | x | 1 | Dzielna s(x) | |
| WXOR | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | Stan Mi | |
| | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | s(x) : H0B | $x^4$ =M17= 1 |
| = WXOR | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | = $s_1$ (x) | |
| | | 1 | 0 | 1 | 1 | 0 | 0 | 0 | $s_1$ (x) : H0B | $x^3$ =M16=1 |
| = WXOR | | 0 | 0 | 1 | 1 | 0 | 1 | 1 | = $s_2$ (x) | |
| | | | | 1 | 0 | 1 | 1 | 0 | $s_2$ (x) : H0B | $x^1$ =M14= 1 |
| = WXOR | | | | 0 | 1 | 1 | 0 | 1 | = $s_3$ (x) | |
| | | | | | 1 | 0 | 1 | 1 | $s_3$ (x) : H0B | $x^0$ =M13= 1 |
| = | | | | | 0 | 1 | 1 | 0 | = r(x) | |

Rys. 10. Dwie wersje programu dzielenia wielomianów: a) zmienna długość dzielnika; b) stała długość dzielnika; c) i d) – przedstawienie działania operacyjnego każdej z wersji

wpisać słowo danych h(x) do markerów M13÷M17, przez co w rejestrze markerów M10÷M17 uzyskuje się poszukiwane słowo kodowe.

### 3.4. Kodowanie wielomianowe w sterowniku FX3U

Prezentowane programy na rys. 10 a i 10 b stanowią wzorce do tworzenia kodów o większych długościach słowa, w szczególności gdy $n = 8*m$, gdzie m jest liczbą bajtów wielomianu (słowa) kodowego. Programy te mogą być realizowane w każdym typie sterownika FX. W najszybszym ze sterowników serii FX o symbolu FX3U wprowadzono specjalizowaną instrukcję o symbolu CRC (*Cyclic Redundancy Check*) obliczania nadmiaru kodowego, w postaci reszty z dzielenia przez jeden ze znormalizowanych wielomianów generujących o symbolu CRC16, który posiada następującą postać:

$$CRC16 = x^{16} \oplus x^{15} \oplus x^2 \oplus 1 \qquad (4)$$

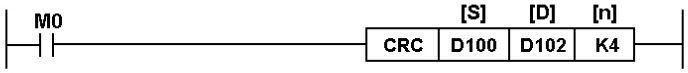
Instrukcja CRC wykonuje dzielenie na zadeklarowanym bloku danych (analogicznie jak instrukcja CDC), a obliczoną 16-bitową resztę wpisuje do zadeklarowanego [D] rejestru. Zapis instrukcji na 4-bajtowym bloku danych (D100 i D101) dla operacji na całych słowach (M8161 = „0”) przedstawiono na rys. 11 [4]. Działanie instrukcji dla tego przypadku obrazuje górna część tabeli danych źródłowych i wyniku CRC (D102).

Dla przypadku wartości markera M8161 = „1” blok danych określony jest przez młodsze bajty zadeklarowanych słów bufora źródłowego, stąd w deklaracji instrukcji dla tego przypadku, przedstawionym w dolnej części tablicy na rys. 11, rejestr wyniku musiałby mieć zapis [D] = D104. Młodszy bajt obliczanej reszty umieszczany jest w rejestrze następnym D105.

Taki sposób deklaracji umieszczenia wyniku dzielenia zaraz za blokiem danych źródłowych pozwala na wykorzystanie rejestrów używanych przez instrukcję CRC również jako bufora nadawczego.

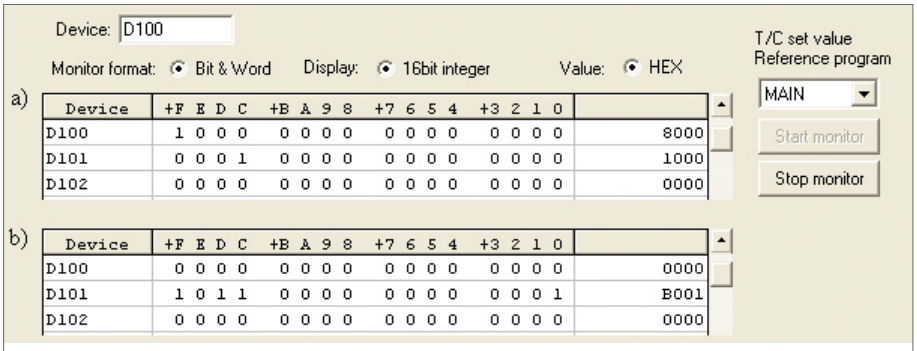
W tabeli na rys. 11 pokazano numerację pozycji (bitów) poszczególnych bajtów słowa kodowego w zapisie wielomianowym, czyli określenie wartości wykładnika m+j poszczególnych składników $x^{m+j}$ wielomianu kodowego. Przyjęto tu zasadę, że pierwszy nadawany bit (d0) bajtu

**CRC: obliczanie 16-bitowego wielomianowego nadmiaru kodowego w rejestrze celu [D] jako reszty z dzielenia wielomianu danych zawartych w stosie [n] bajtów o adresie początkowym [S] przez wielomian generujący CRC16 = $x^{16} \oplus x^{15} \oplus x^2 \oplus 1$**




| Źródło (S) | | Bajt starszy H | | | | | | | | Bajt młodszy L | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Pozycja Di; | i: | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Składnik $x^{m+j}$; | j: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | |
| D100 | m=32 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | |
| D101 | m=16 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | |
| CRC:D102 | m= 0 | **1** | **1** | **1** | **0** | **1** | **0** | **1** | **1** | **0** | **0** | **1** | **1** | **0** | **0** | **1** | **1** | **M8161 = „0"** |
| Składnik $x^{m+j}$; | j: | | | | | | | | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| D100 | m=40 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | |
| D101 | m=32 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | |
| D102 | m=24 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | |
| D103 | m=16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | **M8161 = „1"** |
| $CRC_H$:D104 | m= 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **0** | **0** | **1** | **1** | **0** | **0** | **1** | **1** | |
| $CRC_L$:D105 | m= 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **1** | **1** | **1** | **0** | **1** | **0** | **1** | **1** | |

Rys. 11. Zakresy działania instrukcji CRC dla dwóch stanów markera systemowego M8161




Rys. 12. Tabele rejestracji wyników instrukcji CRC dla wielomianów danych s(x) = 1 i s(x) = 0

danych w transmisji szeregowej jest najstarszą pozycją ($x^7$ dla bajtu, $x^{15}$ dla słowa) w zapisie wielomianowym. Miałoby to istotne znaczenie w odbiorniku transmisyjnym, w którym zastosowanoby dekoder szeregowy, dzielący odbierany strumień danych przez wielomian CRC16.

Aby upewnić się, że instrukcja CRC generuje resztę wg wielomianu CRC16 zgodnie z regułą, która miałaby być zaimplementowana w układzie projektowanego odbiornika (poza systemem sterownika FX3U), należy sprawdzić sposób generacji tej reszty. Jeśli założyć, że słowo danych będzie słowem 16-bitowym, to instrukcja CRC ma za zadanie określenie nadmiaru kodowego dla kodu $(n, k) = (32, 16)$. Jeśli w takim kodzie założyć, że tylko pozycja $x^{16} = 1$, to nadmiar kodowy (reszta z dzielenia) powinien posiadać pozostałe składniki wielomianu CRC16, czyli $r(x) = x^{15} \oplus x^2 \oplus 1$.

Na rys. 12 w tabeli a) pokazano wynik takiego eksperymentu (rysunek przedstawia fragment okna monitora urządzeń oprogramowania narzędziowego GX–Developer dla sterowników FX), w którym w instrukcji CRC zadeklarowano liczbę bajtów danych $k = 2$; bajty rejestru D100. Pozycja $x^{16}$ słowa danych w rejestrze D100 jest równa „1” (pozycja d15; patrz tabela na rys. 11), natomiast resztę z dzielenia przedstawia stan rejestru D101. Pozycja „1” w tym rejestrze odpowiada składnikowi $x^3$ wielomianu reszty, co jest niezgodne z kombinacją oczekiwaną. W wyniku tego kodowania otrzymano wielomian $y1(x) = x^{16} \oplus x^3$.

Zachodzi pytanie, czy kod generowany przez instrukcję CRC jest kodem liniowym w myśl zapisu (1), czyli czy spełniony jest aksjomat A1 z punktu 2. W tym celu zadeklarowano wykonanie tego samego programu z zerowym słowem danych w rejestrze D100. Wynik rejestracji obliczonej reszty przez instrukcję CRC przedstawiono w rejestrze D101 w tabeli b) na rys. 12. Zamiast wyniku zerowego otrzymano wielomian $r1(x) = x^{15} \oplus x^3 \oplus x^2 \oplus 1$, co sugeruje, że kod generowany przy pomocy instrukcji CRC nie jest kodem liniowym.

Jedyną alternatywą mogącą zaprzeczyć temu stwierdzeniu jest przyjęcie zało-

żenia, że kod generowany przez instrukcję CRC jest kodem warstwowym kodu liniowego. Kod warstwowy otrzymuje się z kodu liniowego przez dodanie (mod. 2) do każdego słowa kodowego słowa o stałej kombinacji, tzw. słowa tworzącego warstwę. Tym słowem może być wielomian niezerowej reszty $r1(x)$, otrzymanej w wyniku dzielenia zerowego słowa danych. Dodanie tego wielomianu do określonego wyżej wielomianu kodowego $y1(x)$ daje następujący wynik:

$$y1(x) \oplus r1(x) = (x^{16} \oplus x^3) \oplus (x^{15} \oplus x^3 \oplus x^2 \oplus 1) = \\ = x^{16} \oplus x^{15} \oplus x^2 \oplus 1 = \text{CRC16} \quad (5)$$

Oznacza to, że instrukcja CRC pozwala kodować dane wg wielomianu CRC16, ale warstwowo, czyli z dodaniem słowa o stałej kombinacji do każdego słowa – wielomianu kodowego.

## 4. Podsumowanie

Z przytoczonych przykładów w podrozdziale 3 wynika, że w każdym typie sterownika z rodziny FX można zastosować dowolny sposób kodowania i dekodowania nadmiarowego danych. Nawet jeśli sterownik nie posiada specjalizowanej do tego celu instrukcji (np. CRC), to można zastąpić ją procedurą bazującą na instrukcjach uniwersalnych. Tworzenie takiej procedury jest procesem bardziej skomplikowanym i wymagającym znacznej więcej wiedzy od programisty ale jest to zawsze możliwe. Natomiast jeśli w systemie sieciowym występują stacje ze sterownikami z grupy FX3U, to wskazane byłoby w stacjach tego systemu zastosowanie sposobu kodowania opartego na instrukcji CRC, bazującej na wielomianie generującym CRC16. Implikuje to wówczas konieczność wprowadzenia w pozostałych stacjach sieciowych programowej procedury generowania nadmiaru do przesyłanego bloku danych (np. wg rys. 10) wg stosowanej w instrukcji CRC reguły kodowania warstwowego.

## Literatura


[1] Dróżdż J.: *Podstawy kodowania nadmiarowego*. WPP 1980.

[2] Kubiak Z., Mielcarek R.: *Problematyka kodu optymalnego dla zabezpieczenia danych transmisyjnych w Radiotelefonicznym Systemie Telemechaniki*. Prace II MKNT pt.: REAL-TIME SYSTEMS, s. 394–402, Szklarska Poręba, wrzesień 1995.

[3] Mielcarek R.: *Programowanie sterowników PLC. Przewodnik do ćwiczeń laboratoryjnych*. WPP 2012.

[4] Mitsubishi Electric: PROGRAMMING MANUAL – Basic & Applied Instrictions Edition. FX3u/FX3UC Series Programmable Controllers. 2005.

[5] Mitsubishi Electric: MELSEC FX Series. User's Manual (Data Communication) FX1S/FX1N/FX2N(C)/FX3U Interface Modules. 2007.



Roman Mielcarek – Politechnika Poznańska, Instytut Informatyki, e-mail: roman.mielcarek@cs.put.poznan.pl



artykuł recenzowany